Yujian Li (yl7kd)
De Ouyang (do5xb)

Describe your basic path finding algorithm. Show a brief analysis of how well it works on a few different datasets that you produced. What kinds of dataset are more inefficient? Why is that the case?
Describe how you adapt your algorithm when dealing with uncertain situations. How did you deal with the fact that robots sometimes incorrectly view the space in the world?

Produce data that shows how well your algorithm performs on different inputs. What happens if you slightly tweak or change your algorithm? How do these changes affect the performance and why?

We used A* search with evaluation function f(N) = g(N) + h(N). g(N) represents the number of steps from the initial point to the current point. h(N) represents the Chebyshev's distance from the current point to the goal. Prioritizing nodes that have lower value from the evaluation function, we return the first path that reaches the goal or return null if the goal is unreachable.

Overall our algorithm is very efficient. When the goal is further from the initial position, the number of pings we need is higher and hence our algorithm gets less efficient. It is reasonable because the robot only moves one step a time and to move to the next step we ping the surrounding positions.

In the certain case, we only ping the positions that are 1 step from robot's current position and hence the number of pings is low.

To deal with uncertain situations, we adapt our original algorithm such that we ping 501 times and go with the higher probability outcome instead of 1 time to determine whether a wall lies in a certain position and save the result for that position so we don't make redundant pings in the future. Since we only pinged positions that are 1 step away from the robot, it is reasonable to assume that the probability of the robot getting the correct view is higher than it getting the incorrect view.

As for the efficiency of our algorithm under uncertain situations, we have been able to keep it reasonably efficient while ensuring accuracy in determining what a position is. But it is less efficient compared to that in algorithm due to presence of uncertainty.

In general, bigger data size lowers our algorithm's efficiency in both certain and uncertain situations because we will have to ping more times. The effect is more significant on the uncertain situations because we ping 501 times to determine what lies in a position.

At first we didn’t record the position we already determined. So for the first input with uncertainty, the number of pings was 9025. Then we changed our algorithm to create a 2D array to record the positions and that cut down the number of pings by at least a factor of two to 4015. And then we noticed that we didn’t check whether the neighboring positions are the goal before we ping 501 times. We then changed the algorithm to check beforehand and cut down the number of pings by more than 500 times to 3503. These changes improve the performance by reduce redundancy in checking the positions. It is a tradeoff between space and time. Hence in the case of certain situations. The total number of pings wasn’t too high and we only need one ping to check one position and hence we didn’t save the checked positions.

After all the optimization, for input2, the number of pings is 14529; for input3, the number of pings is 47595 and for input4 the number of pings is 102204.